REMARKS

Initially, Applicant expresses appreciation to the Examiner for the courtesies extended in the recent telephonic discussion held regarding this application with Applicant's representative. The amendments and remarks presented herein are consistent with the discussions. Accordingly, entry of this amendment and reconsideration of the pending claims is respectfully requested.

The Office Action, mailed January 19, 2007, considered and rejected claims 1-10. Claims 1-10 were rejected under 35 U.S.C. § 101 as being directed to non-statutory subject matter. Additionally, Claims 1-10 were rejected under 35 U.S.C. § 103(a) as being unpatentable over *Birsan* (U.S. Patent No. 6,848,078) in view of *Graunke* (U.S. Patent No. 5,852,826), and further in view of *Al-Shaer* ("A Survey of Event Filtering Mechanisms for Dynamic Multi-point Applications").[1]

By this paper, claims 1, 6 and 7 have been amended, claims 4 and 5 cancelled, and claim 11.[2] Accordingly, following this amendment, claims 1-3 and 6-11 are pending, of which claims 1 and 11 are the only independent claims at issue.

As reflected in the above claim listing, the claims generally relate to methods and computer readable storage media for reporting events that occur within or which relate to a computer system. As reflected in independent claim 1, for example, detected events are selectively reported according to filtering trees which are merged to create an output filter tree which, when traversed, determines which events to selectively report. As reflected in claim 1, for example, first and second filtering trees are accessed that are usable for determining whether an event should be reported to a subscriber application. It is then determined that a first node at the topmost level of a first filtering tree and at the topmost level of a second filtering tree are both OR nodes. Thereafter, and in response to determining that an OR node is at the topmost level of the first and second filtering trees, a single output tree is created and stored on computer readable storage media. The created and stored output filtering tree provides a resulting OR node at its topmost level. Additionally, the method then includes adding each child node of the first and second trees into the output tree as child nodes, such that the single output tree, when

[1] Although the prior art status of the cited art is not being challenged at this time, Applicant reserves the right to challenge the prior art status of the cited art at any appropriate time, should it arise. Accordingly, any arguments and amendments made herein should not be construed as acquiescing to any prior art status of the cited art.

[2] Support for the above claim amendments and new claims can be found throughout Applicant's original application, and the references incorporated by reference. For example, among other places, the amended claims have support in pages 22-26 of the original application, and in the originally filed claims.

traversed with actual event data, is configured to identify when an event corresponding to the actual event data is to be reported. The act of adding each child node further includes merging each child node of the first tree with a child node of the second tree into a merged child node when the nodes can be successfully combined. Where nodes cannot be successfully combined, each child node is added to the output tree as a child of the resulting OR node. Further, after any child of the first and second filtering trees have been added to the output filtering tree, they are eliminated from further merging.

While the cited references generally relate to the use of computing systems to combine resources into hierarchical trees, Applicant respectfully submits that they fail to disclose or suggest each and every limitation of the pending claims. For example, among other things, the cited references fail to disclose or suggest first, second and merged filtering trees having a resulting OR node at a topmost level, or wherein upon adding a child node to a resulting tree, such node is eliminated from further consideration for merging, as recited in combination with the other claim elements.

For example, *Birsan* generally relates to an XML comparison system in which an XML base file is compared with a modified version of the base file. (*Abstract*; Col. 1, ll. 39-44). Specifically, when the elements are compared, a tree structure is the provided which combines elements of the base file and the modified file, and highlights the portions which are different between the files. (Col. 1, ll. 43-48).

Before the file can be used, the user must resolve the differences between the two files. (Col. 1, ll. 48-51). For example, when an element is added in the modified file, the user can select whether the new element should be incorporated into a merged file. (Col. 1, ll. 54-59) Similarly, if an element was removed in the modified file, the user selects whether the removed element should or should not be included in the merged file. (Col. 1, ln. 67 to Col. 2, ln. 5). For modified elements which conflict, the user selects whether to include the old information or the new information in the merged file. (Col. 1, ll. 59-67). After the user has resolved the differences, the merged file is created containing the elements as selected by the user from the base and modified files. (Col. 1, ll. 48-51). Thus, *Birsan* discloses that elements added to the merged file for use can include added elements, can include removed elements, and can include an element corresponding to an old or new version of an element, as selected by the user. In other words, *Birsan* discloses that new and old elements are separate elements, whereas for changed elements, only the new or old information—but not both—is included in the merged file. Thus, *Birsan* discloses that where information could be merged for a child node, the system instead selects the value from the base file or the modified file, such that node information from only one of the files is included. In other words, while a merged file is created from the base and modified files, any particular merged node contains elements from only one of the base or modified files. Thus, *Birsan* discloses that where nodes could be merged, one or the other is selected, rather than merged, as claimed in combination with the other claim elements. Further, *Birsan* has no disclosure regarding filtering events for selective reporting, reading or creating OR nodes, or adding nodes such that when added they are eliminated from further consideration for merging, as claimed in combination with the other claim elements.[3]

Graunke is generally directed to a parallel sorting technique. As disclosed in *Graunke*, for example, a stream of records are provided as an input data set, and written into a series of input buffers. (Col. 5, ll. 32-38). The stream elements are marked and inserted into a merge tree. (Col. 6, ll. 34-40). In particular, to merge the stream elements, two elements are paired at a leaf node, and merged at the node to produce a stream element output as a higher level stream document. (Col. 6, ll. 63-66). This continues for each set of leaf nodes until no more stream elements are to be paired. (Col. 6, ln. 66 to Col. 7, ln. 2). Then the leaf nodes, as embodied in the merged stream elements, are further merged into an even higher node 34. (Col. 8, ll. 9-28). In other words, merging in the tree construction runs from the bottom of the tree upward, such that stream elements are merged multiple times into higher-and-higher level stream documents. (*See, e.g.,* Fig. 3 where SEs 30a and 30b are merged into SE 36a and then subsequently into SEs 40a and 44).

Accordingly, in direct contrast to the present invention, in which mergeable nodes are merged and then eliminated from consideration for merging, *Graunke* discloses that the nodes continue to be merged. (Office Action, p. 3). Further, *Graunke* is directed to sorting algorithms, and fails to have any disclosure relating to an OR node, an OR node at a topmost level of an output filtering tree, or any filtering tree usable to selectively report computing events.

[3] Inasmuch as nodes in *Birsan* are exclusive, such that they cannot be merged into the final tree, *Birsan* not only fails to disclose or suggest each element of the pending application, but modification of *Birsan* to include merging nodes from two different trees into a single node would change the principles of operation on which *Birsan* is based, and render it inoperative for its intended use. For example, as described in *Birsan* nodes from two trees which relate to the same information but which contains conflicting information cannot be merged as it would render the system inoperative to have two equally viable but contradictory sets of available information for use.

The *Al-Shaer* reference is the only reference which has any relation to filtering computing events to determine significant occurrences that are reported by notification messages. (*Abstract*; Section 2.1). In identifying events, an event filter is set-up which is based on a Boolean-value true/false expression. (Section 2.1). Predicates may then be joined by AND, OR or NOT operators to compose arbitrarily complex filter expressions. (*Id.*) Filters are, accordingly, predicates joined by operators, and filters can be joined together to form an optimized filter. (*Id.*)

Thus, the limited disclosure in *Al-Shaer* with regard to OR operators is that they are used to connect Boolean predicates, and that the predicates joined by operators form a filter. Thereafter, multiple filters can be joined and optimized by using filter composition. *Al-Shaer* fails to disclose, however, that an OR node is at the top-most level of a first filtering tree, a second filtering tree, and an output filtering tree, as claimed. Moreover, *Al-Shaer* merely disdcloses that filters are joined by using filter composition and fails to disclose any merging of child nodes, as recited in combination with the other claim elements, particularly in any way that eliminates added and merged nodes from further merging consideration.

Accordingly, when considered in their entireties, the cited references fail to disclose or suggest various elements of the pending claims, and thus fail to make obvious the pending claims. For example, among other things, the cited references fail to disclose an OR node at the top most level of any tree, let alone atop each of three filtering trees. The cited references further fail to disclose wherein after a child node has been merged with another child node, and added as a merged child node in an output filtering tree, each such node is removed from further consideration for merging, as cited in combination with the other claim elements. Indeed, *Birsan* expressly discloses that any node contains information from only one of the trees, and that they are not merged, *Graunke* discloses that initial merging is only the first step and that each element is subsequently merged all they way up to a top level node, and *Al-Shaer* merely discloses that filters are combined.[4]

[4] At least in part due to the very different uses of each of the systems of the cited references, Applicant also respectfully submits that even were all the elements of the pending claims taught, there would be no motivation to combine the references to obtain the claimed invention. For example, *Birsan* relates to comparing document versions, *Graunke* relates to sorting multiple different documents, and *Al-Shaer* relates to notifying computing systems of events. While each system may use hierarchical trees, the structure and content of the trees is highly different, such that there is no apparent cross-over in these systems. For example, the tree of one process could not be used to adequately operate in any of the other disclosed manners, and particularly not in any way that is enabled or that would lead one of ordinary skill in the art to consider implementing aspects of each to obtain the claimed invention. Moreover, in *Birsan* the top level of the trees is the name of an XML class. Changing the XML class to an OR node would alter the functionality of the entire tree and render it inoperative for its intended use. For example, each class has multiple

In view of the foregoing, Applicant respectfully submits that the other rejections to the claims are now moot and do not, therefore, need to be addressed individually at this time. It will be appreciated, however, that this should not be construed as Applicant acquiescing to any of the purported teachings or assertions made in the last action regarding the cited art or the pending application, including any official notice. Instead, Applicant reserves the right to challenge any of the purported teachings or assertions made in the last action at any appropriate time in the future, should the need arise. Furthermore, to the extent that the Examiner has relied on any Official Notice, explicitly or implicitly, Applicant specifically requests that the Examiner provide references supporting the teachings officially noticed, as well as the required motivation or suggestion to combine the relied upon notice with the other art of record.

In the event that the Examiner finds remaining impediment to a prompt allowance of this application that may be clarified through a telephone interview, the Examiner is requested to contact the undersigned attorney by telephone at (801) 533-9800.

Dated this 18th day of April, 2007.

Respectfully submitted,

[illegible]

RICK D. NYDEGGER
Registration No. 28,651
JENS C. JENKINS
Registration No. 44,803
COLBY C. NUTTALL
Registration No. 58,146
Attorneys for Applicant
Customer No. 047973

RDN:JCJ:CCN:gd
GD0000001724V001

subclasses and attributes which are coextensive, and not alternatives. Changing the class to an OR node would thereby eliminate the use of multiple attributes to describe an object as expressly taught in *Birsan* and as desired for XML statements, and would require a complete change of XML which relies on class names to recognize and use objects. Similarly, in *Graunke*, the top level node is content in the form of a merged set of documents. To change document data to an OR node would alter the entire functionality and utility of the system in that it would remove the content from the top node.